

RP HW

Problem1 - Obtain and plot the values of $R_{ss}[m], \forall 0 \leq m \leq 20$, in Figure 1.

By Yule-Walker equations , We can get

$$\begin{aligned} R_{ss}[0] + a_1 R_{ss}[1] + a_2 R_{ss}[2] + a_3 R_{ss}[3] + \cdots + a_{20} R_{ss}[20] &= b_0^2 \\ R_{ss}[1] + a_1 R_{ss}[0] + a_2 R_{ss}[1] + a_3 R_{ss}[2] + \cdots + a_{19} R_{ss}[20] &= 0 \\ \vdots \\ R_{ss}[20] + a_1 R_{ss}[19] + a_2 R_{ss}[18] + a_3 R_{ss}[17] + \cdots + a_{20} R_{ss}[0] &= 0 \end{aligned}$$

Calculate and plot the values of $R_{ss}[m]$ for $0 \leq m \leq 20$ by substituting the values of a_1, a_2, \dots, a_{20} , and b_0 into the equation.

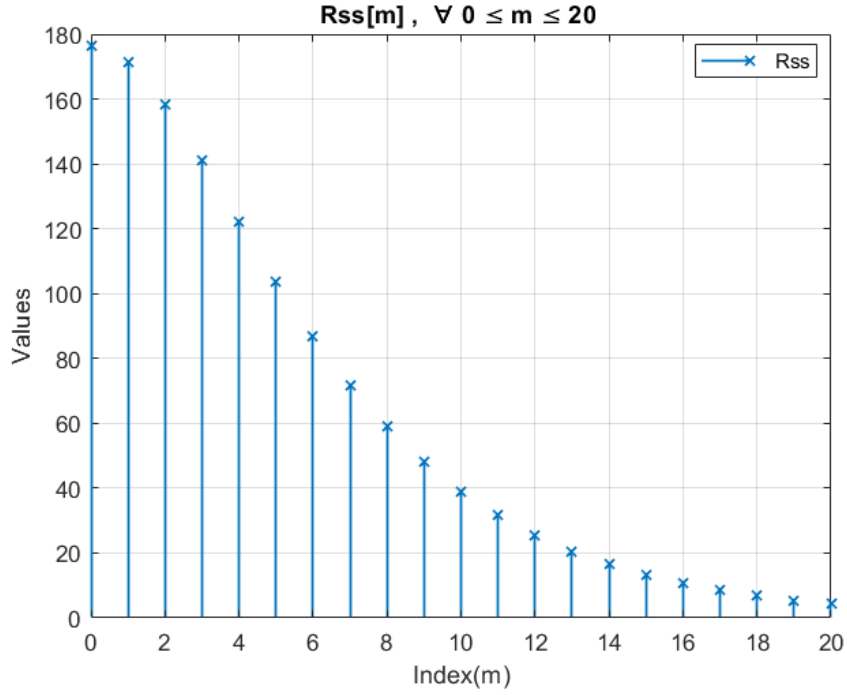


Figure 1: Rss

```

1  noiseLength = 1000000; % length s[n],n=[1:1000000]
2  s=zeros([1 noiseLength+3]); % s initial value
3  a=[1,-1.9,1.18,-0.24]; % mean a0,a1,a2,a3
4  b0=2;
5  % set white Gaussian noise value
6
7  meanValue = 0; % mean value
8  whiteNoise = meanValue + randn(1, noiseLength);
9  varianceValue = var(whiteNoise);
10 disp('White Noise Variance :');
11 disp(varianceValue);
12
13 % find s[n] value
14 % Note matlab idx from 1 not 0
15 for i=4:noiseLength+3
16     s(i)=b0*whiteNoise(i-3)-a(2)*s(i-1)-a(3)*s(i-2)-a(4)*s(i-3);
17 end
18 s=s(4:end); % remove init point
19
20 M=20;
21 % solve 4-order equation at first
22 % Ax=b find x , where x is Rss[0]~Rss[3]
23 A=[a(1),a(2),a(3),a(4);
24     a(2),a(1)+a(3),a(4),0;
25     a(3),a(2)+a(4),a(1),0;
26     a(4),a(3),a(2),a(1)];
27 b=[b0^2;0;0;0];
28 Rss=transpose(A\b);
29 disp('Rss[0] ~ Rss[3]:');
30 disp(Rss);
31 Rss=[Rss,zeros([1,17])];
32 % solve Rss[4]~Rss[20]
33 for i=5:M+1
34     % Rss[4(i)]=-a1*Rss[3(i-1)]-a2*Rss[2(i-2)]-a3*Rss[1(i-3)]
35     Rss(i)=-a(2)*Rss(i-1)-a(3)*Rss(i-2)-a(4)*Rss(i-3);
36 end

```

Listing 1: Problem1 Matlab Code

Problem2 - plot $\{\hat{s}_{20}[n], \forall 10000 \leq n \leq 10100\}$ and $\{s[n], \forall 10000 \leq n \leq 10100\}$, in Figure 2

According to the definition of $\hat{s}_M[n] = \hat{E}\{s[n] \mid s[n-k], 1 \leq k \leq M\}$ and substituting $M = 20$, $\forall 10000 \leq n \leq 10100$ into the definition, we can get

$$\hat{s}_{20}[n] = \hat{E}\{s[n] \mid s[n-k], 1 \leq k \leq 20\}$$

Based on theorem 13-1, we know that $E[(s[n] - \sum_{k=1}^N h[k]s[n-k])s[n-m]] = 0, \forall 1 \leq m \leq 20$. According to the result obtained from the previous equation, We can determine the value of $h[k]$, where $\forall 1 \leq k \leq 20$, by solving the following system of simultaneous equations:

$$\begin{aligned} R_{ss}[1] - \sum_{k=1}^N h[k]R_{ss}[1-k] &= 0 \\ R_{ss}[2] - \sum_{k=1}^N h[k]R_{ss}[2-k] &= 0 \\ &\vdots \\ R_{ss}[20] - \sum_{k=1}^N h[k]R_{ss}[20-k] &= 0 \end{aligned}$$

Finally, we can calculate the $\hat{s}_{20}[n] = \sum_{k=1}^N h[k]s[n-k], \forall 10000 \leq n \leq 10100$

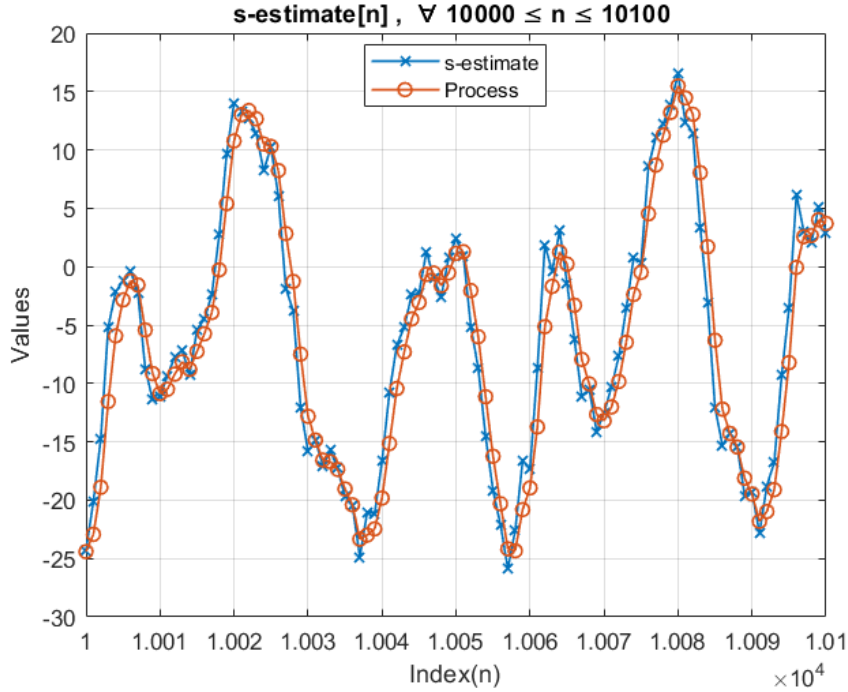


Figure 2: $\hat{s}_{20}[n], \forall 10000 \leq n \leq 10100$

```
1 b=transpose(Rss(2:21));
2 A=[];
3 for i=1:M
4     tmp=flip(Rss(1:i));
5     if M-i>0
6         tmp=[tmp,Rss(2:M-i+1)];
7     end
8     A=[A;tmp];
9 end
10
11 if transpose(A)==A
12     disp('Problem2 - A matrix is same');
13 end
14 h=transpose(A\b);
15
16 s_estimate = [];
17 for i=n
18     tmp=h.*flip(s(i-M+1:i));
19     s_estimate=[s_estimate,sum(tmp)];
20 end
```

Listing 2: Problem2 Matlab Code

Problem3 - $\hat{P}_{20}[n], \forall n \in \{10, 10^2, 10^3, 10^4, 10^5, 10^6\}$ in Figure 3

Based on definition $\hat{P}_M[n] = \frac{1}{n} \sum_{k=1}^n (\hat{s}_M[k] - s[k])^2$, we can directly calculate the error between predicted values and actual values.

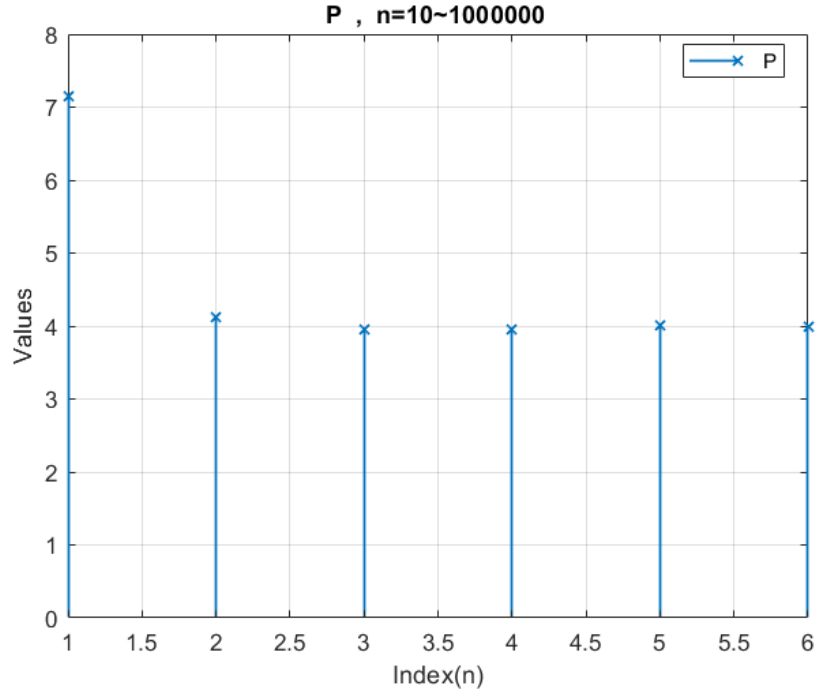


Figure 3: $\hat{P}_{20}[n], \forall n \in \{10, 10^2, 10^3, 10^4, 10^5, 10^6\}$

```

1 %% Count all s_estimate
2 s_estimate=zeros([1 noiseLength]); % clean all
3 for n=1:M
4     tmp=0;
5     for k=1:M
6         if n-k>0
7             tmp=tmp+h(k)*s(n-k);
8         end
9     end
10    s_estimate(n)=tmp;
11 end
12
13 for n=M+1:noiseLength
14     s_estimate(n)=sum(h.*flip(s(n-M:n-1)));
15 end
16 %% Problem-3 n=10-1
17 p_10=0;
18 n_10=10;
19 for i=1:n_10
20     tmp=(s_estimate(i)-s(i))^2;
21     p_10=p_10+tmp;
22 end
23 p_10=p_10/10;
24 d
25 %% Problem-3 n=100-2
26 p_100=0;
27 n_100=100;
28 for i=n_10+1:n_100
29     tmp=(s_estimate(i)-s(i))^2;
30     p_100=p_100+tmp;
31 end
32 p_100=((p_10*n_10) + p_100)/n_100;
33
34 %% Problem-3 n=1000-3
35 p_1000=0;
36 n_1000=1000;
37 for i=n_100+1:n_1000
38     tmp=(s_estimate(i)-s(i))^2;
39     p_1000=p_1000+tmp;
40 end
41 p_1000=((p_100*n_100) + p_1000)/n_1000;
42
43 %% Problem-3 n=10000-4
44 p_10000=0;
45 n_10000=10000;
46 for i=n_1000+1:n_10000
47     tmp=(s_estimate(i)-s(i))^2;
48     p_10000=p_10000+tmp;
49 end
50 p_10000=((p_1000*n_1000) + p_10000)/n_10000;
51 %% Problem-3 n=100000-5
52 p_100000=0;
53 n_100000=100000;
54 for i=n_10000+1:n_100000
55     tmp=(s_estimate(i)-s(i))^2;
56     p_100000=p_100000+tmp;
57 end
58 p_100000=((p_10000*n_10000) + p_100000)/n_100000;
59 %% Problem-3 n=1000000-6
60 p_1000000=0;
61 n_1000000=1000000;
62 for i=n_100000+1:n_1000000
63     tmp=(s_estimate(i)-s(i))^2;
64     p_1000000=p_1000000+tmp;
65 end
66 p_1000000=((p_100000*n_100000) + p_1000000)/n_1000000;

```

Listing 3: Problem3 Matlab Code

Problem4 - Consider the case in which $\{i[n], -\infty < n < \infty\}$ is a white noise process but $i[n]$ has the following probability density function for each $n \in \mathbb{Z}$.

$$f_{i[n]}(t) = \begin{cases} 0.5e^{-t}, \forall t > 0 \\ 0.5e^t, \forall t < 0 \end{cases}$$

Plot $\hat{P}_{20}[n], \forall n \in \{10, 10^2, 10^3, 10^4, 10^5, 10^6\}$, in Figure 4

Based on definition $\hat{P}_M[n] = \frac{1}{n} \sum_{k=1}^n (\hat{s}_M[k] - s[k])^2$, we can directly calculate the error between predicted values and actual values.

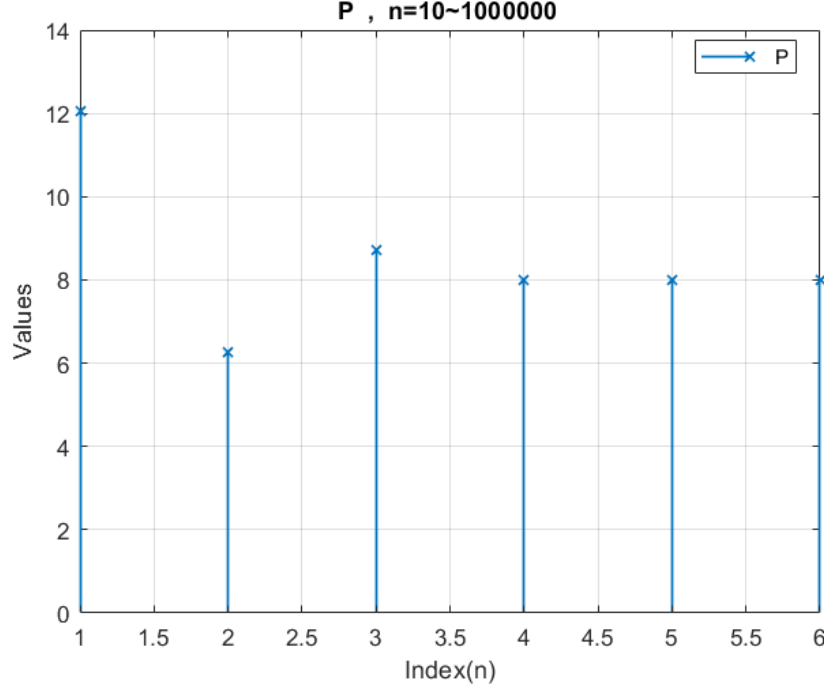


Figure 4: $\hat{P}_{20}[n], \forall n \in \{10, 10^2, 10^3, 10^4, 10^5, 10^6\}$

```

1 %% create Laplace distribution noise
2 noiseLength=1000000;
3 s=zeros([1 noiseLength+3]); % s initial value
4 a=[1,-1.9,1.18,-0.24]; % mean a0,a1,a2,a3
5 b0=2;
6
7 m=1;
8 mu = 0;
9 sigma = sqrt(2);
10 u = rand(m, noiseLength)-0.5;
11 b = sigma / sqrt(2);
12 laplaceRandomNoise = mu - b * sign(u).* log(1- 2* abs(u));
13
14 for i=4:noiseLength+3
15     s(i)=b0*laplaceRandomNoise(i-3)-a(2)*s(i-1)-a(3)*s(i-2)-a(4)*s(i-3);
16 end
17
18 s=s(4:end); % remove init point

```

Listing 4: Problem4 - Create Laplace distribution noise

```

1 %% Count Rss
2 M=20;
3 % Ax=b find x , x is Rss[0]~Rss[3]
4 A=[a(1),a(2),a(3),a(4);
5     a(2),a(1)+a(3),a(4),0;
6     a(3),a(2)+a(4),a(1),0;
7     a(4),a(3),a(2),a(1)];
8 b=[b0^2;0;0;0];
9 Rss=transpose(A\b);
10 disp('Rss[0] ~ Rss[3]:');
11 disp(Rss);
12 Rss=[Rss,zeros([1,17])];
13 % solve Rss[4]~Rss[20]
14 for i=5:M+1
15     % Rss[4(i)]=-a1*Rss[3(i-1)]-a2*Rss[2(i-2)]-a3*Rss[1(i-3)]
16     Rss(i)=-a(2)*Rss(i-1)-a(3)*Rss(i-2)-a(4)*Rss(i-3);
17 end

```

Listing 5: Problem4 - Count Rss

```

1 %% Count all s_estimate
2 s_estimate=zeros([1 noiseLength]); % clean all
3 for n=1:M
4     tmp=0;
5     for k=1:M
6         if n-k>0
7             tmp=tmp+h(k)*s(n-k);
8         end
9     end
10    s_estimate(n)=tmp;
11 end
12
13 for n=M+1:noiseLength
14     s_estimate(n)=sum(h.*flip(s(n-M:n-1)));
15 end

```

Listing 6: Problem4 - Count all \hat{s}

```

1  %% Problem-4  n=10-1
2  p_10=0;
3  n_10=10;
4  for i=1:n_10
5      tmp=(s_estimate(i)-s(i))^2;
6      p_10=p_10+tmp;
7  end
8  p_10=p_10/10;
9  disp('p_10 :');
10 disp(p_10);
11 %% Problem-4  n=100-2
12 p_100=0;
13 n_100=100;
14 for i=n_10+1:n_100
15     tmp=(s_estimate(i)-s(i))^2;
16     p_100=p_100+tmp;
17 end
18 p_100=((p_10*n_10) + p_100)/n_100;
19 disp('p_100 :');
20 disp(p_100);
21 %% Problem-4  n=1000-3
22 p_1000=0;
23 n_1000=1000;
24 for i=n_100+1:n_1000
25     tmp=(s_estimate(i)-s(i))^2;
26     p_1000=p_1000+tmp;
27 end
28 p_1000=((p_100*n_100) + p_1000)/n_1000;
29 disp('p_1000 :');
30 disp(p_1000);
31 %% Problem-4  n=10000-4
32 p_10000=0;
33 n_10000=10000;
34 for i=n_1000+1:n_10000
35     tmp=(s_estimate(i)-s(i))^2;
36     p_10000=p_10000+tmp;
37 end
38 p_10000=((p_1000*n_1000) + p_10000)/n_10000;
39 disp('p_10000 :');
40 disp(p_10000);
41 %% Problem-4  n=100000-5
42 p_100000=0;
43 n_100000=100000;
44 for i=n_10000+1:n_100000
45     tmp=(s_estimate(i)-s(i))^2;
46     p_100000=p_100000+tmp;
47 end
48 p_100000=((p_10000*n_10000) + p_100000)/n_100000;
49 disp('p_100000 :');
50 disp(p_100000);
51 %% Problem-4  n=1000000-6
52 p_1000000=0;
53 n_1000000=1000000;
54 for i=n_100000+1:n_1000000
55     tmp=(s_estimate(i)-s(i))^2;
56     p_1000000=p_1000000+tmp;
57 end
58 p_1000000=((p_100000*n_100000) + p_1000000)/n_1000000;
59 disp('p_1000000 :');
60 disp(p_1000000);

```

Listing 7: Problem4 - Count all P